CDMI Extension: LTFS Export

Version 1.1a

"Publication of this Working Draft for review and comment has been approved by the Cloud Storage Technical Working Group. This draft represents a "best effort" attempt by the Cloud Storage Technical Working Group to reach preliminary consensus, and it may be updated, replaced, or made obsolete at any time. This document should not be used as reference material or cited as other than a 'work in progress.' Suggestion for revision should be directed to http:/snia.org/feedback."

Working Draft

Revision History

Date	Version	By	Comments
08-19-2014	1.1a	LTFS TWG	New version of LTFS Export extension created for the CDMI 1.1.0 revision of the standard. Derived from the 1.0d draft.

LTFS Export Extension

Overview

LTFS provides a cost-effective and time-efficient approach to bulk loading and transfer of large quantities of stored cloud data. The LTFS Export Extension standardizes how LTFS tape collections can be accessed through CDMI and used for bulk data transfers.

Modifications to the 1.1.0 spec:

1. Insert new entry into clause 2

SNIA, 2014, *Linear Tape File System (LTFS) Format Specification 2.2.0* - http://snia.org/sites/default/files/LTFS_Format_2.2.0_Technical_Position.pdf

2. Insert new clause "13.9 LTFS Export"

The Linear Tape File System (LTFS) standard defines an interoperable way to store and exchange files and directories on tape storage devices. LTFS permits files to be stored on one or more LTFS volumes, providing a single namespace view for arbitary sized files and file collections.

LTFS exports allow a collection of CDMI objects to be stored and retrieved from one or more LTFS volumes. An LTFS volume or set of volumes is first associated with a CDMI container, which instructs the storage system to provide access to the contents of those volumes via CDMI. This is accomplished by specifying a CDMI export for that container. Once a CDMI container is associated with LTFS volumes, the container can be used to access or deserialize the contents of the LTFS volumes. In addition, the container can act as a destination for object creation, copying, and serialization via standard CDMI operations.

LTFS exports do not define the underlying mechanisms by which the tapes are accessed, which are implemented by various software layers running below the cloud interface layer and above the tape library. This is left to the LTFS library implementations.

13.9.1 Managing Latency in LTFS Exports

Objects residing on an LTFS export may have significantly higher latency as a consequence of being stored on tape. The CDMI standard provides access to latency information through the cdmi_latency_provided metadata item (see clause 16.5, Table 120 - Provided Values of Data Systems Metadata Items), allowing a client to determine the expected latency for the object before making the request.

If an object has significant latency, the client shall be able to tolerate the specified latency between issuing the request headers and body and receiving the response headers and body.

If supported by the server, changing the cdmi_latency metadata (see clause 16.5, Table 120) associated with one or more objects shall instruct the server to cache the objects in a lower latency location. When the server has cached the data, the cdmi_latency_provided shall reflect the lower latency of the cache.

cdmi_latency can be changed for individual objects, for containers (affecting all contained objects that do not have an explicit cdmi_latency metadata item specified), and for arbitary groups of objects using the CDMI jobs functionality.

If a client intends to wait until objects have lower latency before accessing them, the client may create a CDMI notification queue (see clause 21) to receive notifications of when the provided latency for objects changes. This allows the client to avoid having to poll on the objects.

13.9.2 Associating CDMI Containers and LTFS Volumes

To attach a container to a set of LTFS volumes, the information identifying the LTFS volumes to be associated is specified as export metadata for the container at container creation time, or by modifying an existing empty container to add the required export metadata.

Required members of the protocol structure for LTFS are

- "ltfs_volume_uuids". One or more LTFS Volume UUIDs that are to be used for LTFS filesystem access. If the first volume UUID listed has a valid LTFS Transfer Request XML file in the root directory, the storage system shall automatically update the ltfs_volume_uuids list to include all LTFS Volume UUIDs referenced in the Transfer Request XML file.

Optional members of the protocol structure for LTFS are

- "ltfs_manifest". If present and "true", this indicates that a LTFS Transfer Request XML file shall be created. If absent and an LTFS Transfer Request XML file exists on the LTFS volumes, the storage system shall create this metadata item and set it to the value "true".
- "usermap". As described in 13.2 Exported Protocol Structure, setting a single user map entry of {"myuser", "<-", "*"} will map all existing LTFS users to the "myuser" account.
- "groupmap". As described in 13.2, a single group map entry of {"mygroup", "<-", "*"} will map all existing LTFS groups to the "mygroup" account.
- "domainmap". Uses the same approach described in 13.2. Setting a single domain map entry of {"/cdmi_domains/mydomain/", "<-", "*"} will map all domains on the LTFS export to the "cdmi_domains/mydomain/" CDMI domain.

Example 1: Create a new CDMI container that is associated with four LTFS volumes:

```
PUT /ltfs_exported_container/ HTTP/1.1
X-CDMI-Specification-Version: 1.0.2
Content-Type: application/cdmi-container
Accept: application/cdmi-container

{
   "exports" :  {
      "ltfs" : {
         "ltfs_volume_uuids" : [
             "1F912610-3F48-43F3-A53A-D0761B0238DE",
             "0A198010-740E-433B-920F-0CC95CDD0C7F",
             "5573B072-FFF7-408A-A599-3FC383E72DDC",
             "6DECAAD5-9507-4052-876A-F45B1CE1F2AA"
          ],
         "usermap" : {
            {"myuser", "<-", "*"}
         },
         "groupmap" : {
            {"mygroup", "<-", "*"}
         },
         "domainmap" : {
            {"/cdmi_domains/mydomain/", "<-",
"http://sourcecloud.example.com/cdmi_domains/source/"}
```

```
        }
      }
   }
}
```

If all of the requested volume UUIDs are known to the storage system and are accessible, the container PUT will succeed without errors, and all LTFS files and objects stored on the specified volumes will be accessible through the exported container.

The association shall be removed by deleting the container or deleting the export metadata. Deleting the container shall not delete any stored files or objects.

13.9.3 Storing Data Objects on LTFS

CDMI data objects shall be stored as ordinary LTFS files, with extended attributes as described in 13.9.8 Storing Object Metadata on LTFS.

For each CDMI data object with an object ID, an LTFS symlink with the name of the object ID that points to the LTFS file corresponding to the data object shall be created in an "/cdmi_objectid/" directory in the root directory of the LTFS volume. If a data object only has an ID, the LTFS file is stored directly in the "/cdmi_objectid/" directory.

For example, a data object named "myDataObject" with ID 00007E7F00102E230ED82694DAA975D2 shall be translated into the following files on an LTFS volume:

```
/myDataObject
/cdmi_objectid/00007E7F00102E230ED82694DAA975D2 -> /myDataObject
```

13.9.4 Storing Versioned Data Objects on LTFS

CDMI data objects with versions shall be stored as a collection of LTFS file system items:

- An LTFS symlink to the current version file inside the versioned object directory;
- An LTFS directory, where the name of the directory is the same as the name of the versioned data object, appended with “.cdmi_versions”. This directory includes the extended attributes as described in 13.9.8 Storing Object Metadata on LTFS; and
- A file corresponding to each version of the versioned object in the versioned object directory where each file shall be a valid standalone object. The naming of the version files is up to the implementation.

For example, a versioned data object named “vObj”, with three versions, shall be translated into the following files on an LTFS volume:

```
/vObj -> /vObj.cdmi_versions/vObj;3
/vObj.cdmi_versions/
/vObj.cdmi_versions/vObj;3
/vObj.cdmi_versions/vObj;2
/vObj.cdmi_versions/vObj;1
/cdmi_objectid/00007ED90010849414B876867FC196C8 -> /vObj
/cdmi_objectid/00007E7F0010CEC234AD9E3EBFE9531D -> /vObj.cdmi_versions/vObj;3
/cdmi_objectid/00007E7F0010DCECC805FB6D195DDBCB -> /vObj.cdmi_versions/vObj;2
/cdmi_objectid/00007E7F0010128E42D87EE34F5A6560 -> /vObj.cdmi_versions/vObj;1
```

13.9.5 Storing Container Objects on LTFS

CDMI container objects shall be stored as ordinary LTFS directories, with extended attributes as described in 13.9.8 Storing Object Metadata on LTFS.

For each CDMI container object with an object ID, an LTFS symlink with the name of the object ID that points to the LTFS directory corresponding to the container object shall be created in an "/cdmi_objectid/" directory in the root directory of the LTFS volume.

For example, a container object named "myContainer" would be translated into the following files on an LTFS volume:

```
/myContainer/
/cdmi_objectid/00007E7F00102E230ED82694DAA975D2 -> /myContainer/
```

13.9.6 Storing Queue Objects on LTFS

CDMI queue objects shall be stored as a collection of LTFS file system items:

- An LTFS file representing the queue object;
- An LTFS directory, where the name of the directory is the same as the queue object, appended with ".cdmi_queue". This directory includes the extended attributes as described in 13.9.8 Storing Object Metadata on LTFS; and
- A file corresponding to each enqueued item in the queue, stored inside the queue object directory, where each enqueued item file has a name based on the queueValue number, and shall have extended attributes as described in the second table.

For example, a queue object with two enqueued items would be represented in the following files on an LTFS volume:

```
/myQueue
/myQueue.cdmi_queue/
/myQueue.cdmi_queue/0
/myQueue.cdmi_queue/1
/cdmi_objectid/00007E7F0010CEC234AD9E3EBFE9531D -> /myQueue
```

13.9.7 Storing Reference Objects on LTFS

CDMI reference objects shall be stored as an ordinary LTFS symlink.

For example, a reference object named "ref" pointing to a data object named "bar" shall be translated into the following files on an LTFS volume:

```
/bar
/ref -> /bar
```

13.9.7 Storing Object Fields on LTFS

CDMI fields are stored as LTFS extended attributes according to the following table:

Table 114 – CDMI Fields to LTFS Extended Attribute Mapping

Object Type	Data Object	Versioned Data Object Symlink	Versioned Data Object	Versioned Data Object Version	Container Object	Queue Object	Queue Directory	Queue Item	Reference Object	LTFS Storage Location
objectType										Derived from the LTFS element type: file, directory or symlink
objectID	X		X	X	X	X				Stored as a UTF-8 string containing the CDMI object ID in the LTFS extended attribute ltfs.vendor.cdmi.objectid
objectName	X	X			X	X			X	Derived from the name of the specific LTFS element (file, directory or symlink)
parentURI	X		X	X	X	X				Derived from a concatenation of LTFS directory names separated by the character '/', which represents the position of the object in the LTFS directory structure
parentID	X		X		X	X				Stored as the LTFS extended attribute ltfs.vendor.cdmi.objectid in the parent directory
parentID				X						Stored as a UTF-8 string containing a CDMI object ID in the LTFS extended attribute ltfs.vendor.cdmi.parentid
domainURI	X			X	X	X				Stored as a UTF-8 string containing a domain URI in the LTFS extended attribute ltfs.vendor.cdmi.domainURI
mimetype	X			X				X		Stored as a UTF-8 string containing the MIME type in the LTFS extended attribute ltfs.vendor.cdmi.mimetype
metadata (header)	X			X	X	X				Stored as header/value JSON strings in the LTFS extended attribute ltfs.vendor.cdmi.x-meta

Object Type	Data Object	Versioned Data Object Symlink	Versioned Data Object	Versioned Data Object Version	Container Object	Queue Object	Queue Directory	Queue Item	Reference Object	LTFS Storage Location
Metadata (extended)	X			X	X	X				Stored as CDMI metadata JSON in the LTFS extended attribute ltfs.vendor.cdmi.metadata. Refer to Table 113 - Required Members of the CIFS Protocol Structure for specific metadata mapping.
valuerange	X			X				X		Derived from the LTFS file size
valuetransfer encoding	X			X				X		Stored as a UTF-8 string in the LTFS extended attribute ltfs.vendor.cdmi.valuetransferencoding
value	X			X				X		Derived from the LTFS file content

A list of valid CDMI domains are not transported via LTFS and must be mapped by the storage system.

13.9.8 Storing Object Metadata on LTFS

CDMI metadata is stored as LTFS extended attributes according to the following table:

Table 115 – CDMI Metadata to LTFS Extended Attribute Mapping

Metadata Item	Data Object	Versioned Data Object	Versioned Data Object	Versioned Data Object Version	Container Object	Queue Object	Queue Directory	Queue Item	Reference Object	LTFS Storage Location
cdmi_acl	X		X	X	X	X	X	X		Stored as a standard LTFS ACL metadata (ltfs.permissions.nfsv4acl)

Metadata Item	Data Object	Versioned Data Object	Versioned Data Object	Versioned Data Object Version	Container Object	Queue Object	Queue Directory	Queue Item	Reference Object	LTFS Storage Location
cdmi_size	X			X				X		Derived from the LTFS file length
cdmi_ctime	X		X	X	X	X	X	X		Stored as standard LTFS object metadata (ltfs.createTime)
cdmi_atime	X		X	X	X	X	X	X		Stored as standard LTFS object metadata (ltfs.accessTime)
cdmi_mtime	X		X	X	X	X	X	X		Stored as standard LTFS object metadata (ltfs.modifyTime)
cdmi_acount	X		X	X	X	X	X	X		Stored as a UTF-8 string in the LTFS extended attribute ltfs.vendor.cdmi.acount
cdmi_mcount	X		X	X	X	X	X	X		Stored as a UTF-8 string in the LTFS extended attribute ltfs.vendor.cdmi.mcount
cdmi_hash	X			X						Stored as a UTF-8 string containing the algorithm, length, and value in the formation of "ALGLEN:VALUE" in the LTFS extended attribute ltfs.vendor.cdmi.hash
cdmi_owner	X		X	X	X	X	X	X		Stored as standard LTFS ACL metadata (lfts.permissions.nfsv4acl)
header metadata items	X			X	X	X				Stored as header/value JSON strings in the LTFS extended attribute ltfs.vendor.cdmi.x-meta
All other metadata items	X			X	X	X				Stored as CDMI metadata JSON in the LTFS extended attribute ltfs.vendor.cdmi.metadata

3. Insert into table "Table 104 - Capabilities for Containers"

Capability Name	Type	Description
cdmi_export_container_ltfs	JSON String	If present and "true", this container can be associated with a set of LTFS volumes.